Refine Search

Search Results -

Term	Documents
(12 AND 4).USPT.	4
(L4 AND L12).USPT.	4

Database:
- US Pre-Grant Publication Full-Text Database
- US Patents Full-Text Database
- US OCR Full-Text Database
- EPO Abstracts Database
- JPO Abstracts Database
- Derwent World Patents Index
- IBM Technical Disclosure Bulletins

Search: L13

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Monday, February 07, 2005 Printable Copy Create Case

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT; PLUR=YES; OP=ADJ			
L13	l4 and L12	4	L13
L12	l1 and L11	46	L12
L11	718/102,107.ccls.	1064	L11
L10	l5 and L9	2	L10
L9	dispatch$1	10035	L9
L8	etherealiz$	4	L8
L7	(table$1 or list$) and (object id)	1109	L7
L6	(table$1 ot list$) and (object id)	0	L6
L5	l1 and l3 and L4	32	L5
L4	mutex$ and lock$	358	L4
L3	L2 or (multi adj1 thread$)	3611	L3
L2	multithread$	1495	L2
L1	load$ near3 object$1	16029	L1

PORTAL

US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

loading object + multithreading + mutex + locks

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used loading object multithreading mutex locks

Found **24,686** of **150,138**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Mostly lock-free malloc

Dave Dice, Alex Garthwaite

June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management**, Volume 38 Issue 2 supplement

Full text available: pdf(609.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern multithreaded applications, such as application servers and database engines, can severely stress the performance of user-level memory allocators like the ubiquitous malloc subsystem. Such allocators can prove to be a major scalability impediment for the applications that use them, particularly for applications with large numbers of threads running on high-order multiprocessor systems.This paper introduces Multi-Processor Restartable Critical Sections, or MP-RCS. MP-RCS permits user-level ...

Keywords: affinity, locality, lock-free operations, malloc, restartable critical sections

2 Performance measurements for multithreaded programs

Minwen Ji, Edward W. Felten, Kai Li

June 1998 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems**, Volume 26 Issue 1

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded programming is an effective way to exploit concurrency, but it is difficult to debug and tune a highly threaded program. This paper describes a performance tool called Tmon for monitoring, analyzing and tuning the performance of multithreaded programs. The performance tool has two novel features: it uses "thread waiting time" as a measure and constructs thread waiting graphs to show thread dependencies and thus performance bottlenecks, and it identifies "semi-busy-waiting" points w ...

3 Waiting algorithms for synchronization in large-scale multiprocessors

Beng-Hong Lim, Anant Agarwal

August 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 3

Full text available: pdf(2.72 MB) Additional Information: full citation, abstract, references, citings, index terms

Through analysis and experiments, this paper investigates two-phase waiting algorithms to minimize the cost of waiting for synchronization in large-scale multiprocessors. In a two-phase algorithm, a thread first waits by polling a synchronization variable. If the cost of polling reaches a limit Lpoll and further waiting is necessary, the thread is blocked, incurring an additional fixed cost, B. The choice of Lpoll

Keywords: barriers, blocking, competitive analysis, locks, producer-consumer synchronization, spinning, waiting time

4 Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3
Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, references, citings, index terms

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compilers, dynamic compilation, interpreters, just-in-time compilers

5 Eliminating synchronization bottlenecks in object-based programs using adaptive replication
Martin Rinard, Pedro Diniz
May 1999 **Proceedings of the 13th international conference on Supercomputing**
Full text available: pdf(1.27 MB) Additional Information: full citation, references, citings, index terms

6 Improving server software support for simultaneous multithreaded processors
Luke K. McDowell, Susan J. Eggers, Steven D. Gribble
June 2003 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming**, Volume 38 Issue 10
Full text available: pdf(218.63 KB) Additional Information: full citation, abstract, references, index terms

Simultaneous multithreading (SMT) represents a fundamental shift in processor capability. SMT's ability to execute multiple threads simultaneously within a single CPU offers tremendous potential performance benefits. However, the structure and behavior of software affects the extent to which this potential can be achieved. Consequently, just like the earlier arrival of multiprocessors, the advent of SMT processors prompts a needed re-evaluation of software that will run on them. This evaluation ...

Keywords: runtime support, servers, simultaneous multithreading

7 Eraser: a dynamic data race detector for multithreaded programs
Stefan Savage, Michael Burrows, Greg Nelson, Patrick Sobalvarro, Thomas Anderson
November 1997 **ACM Transactions on Computer Systems (TOCS)**, Volume 15 Issue 4
Full text available: pdf(136.04 KB) Additional Information: full citation, abstract, references, citings, index terms

Multithreaded programming is difficult and error prone. It is easy to make a mistake in synchronization that produces a data race, yet it can be extremely hard to locate this mistake during debugging. This article describes a new tool, called Eraser, for dynamically detecting data races in lock-based multithreaded programs. Eraser uses binary rewriting techniques to monitor every shared-monory reference and verify that consistent locking behavior is observed. We present several case studies ...

Keywords: binary code modification, multithreaded programming, race detection

8 Language support for lightweight transactions
Tim Harris, Keir Fraser
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(224.15 KB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent programming is notoriously difficult. Current abstractions are intricate and make it hard to design computer systems that are reliable and scalable. We argue that these problems can be addressed by moving to a declarative style of concurrency control in which programmers directly indicate the safety properties that they require. In our scheme the programmer demarks sections of code which execute within lightweight software-based transactions that commit atomically and exactly once. Th ...

Keywords: concurrency, conditional critical regions, non-blocking systems, transactions

9 Scalable lock-free dynamic memory allocation
Maged M. Michael
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(213.94 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic memory allocators (malloc/free) rely on mutual exclusion locks for protecting the consistency of their shared data structures under multithreading. The use of locking has many disadvantages with respect to performance, availability, robustness, and programming flexibility. A lock-free memory allocator guarantees progress regardless of whether some threads are delayed or even killed and regardless of scheduling policies. This paper presents a completely lock-free memory allocator. It uses ...

Keywords: async-signal-safe, availability, lock-free, malloc

10 Fast mutual exclusion for uniprocessors
Brian N. Bershad, David D. Redell, John R. Ellis
September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9
Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we describe restartable atomic sequences, an optimistic mechanism for implementing simple atomic operations (such as Test-And-Set) on a uniprocessor. A thread that is suspended within a restartable atomic sequence is resumed by the operating system at the beginning of the sequence, rather than at the point of suspension. This guarantees that the thread eventually executes the sequence atomically. A restartable atomic sequence ...